Lab 3

Due date: Friday March 18th

Instructions:

Import the project to Eclipse
File -> import projects from file system or Archive -> select directory for import source

Add Junit 4 If needed.
Right click java project -> Build Path -> add Libraries -> select Junit 4 -> click finish

1- Understanding the Profile Class

Inspect the profile code and the other classes in the zip file for Lab 3

You can also read pages 17-21 of the recommended lab book if needed
Alternatively, you can review these pages in the PDF document (Lab 3 Book Copy upload)

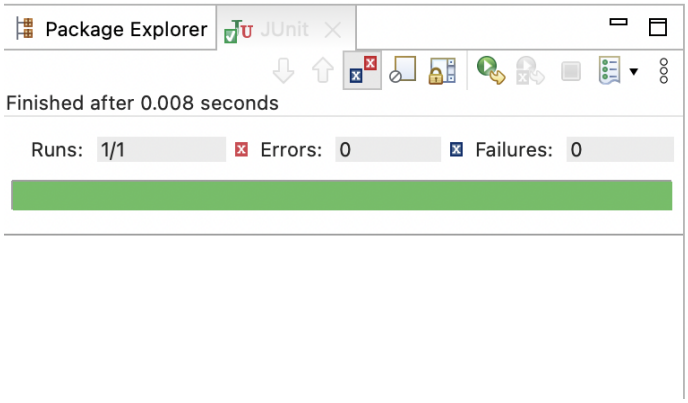
2- Write a test in ProfileTest to cover one path of matches function

We need a Profile instance and a Criteria object to pass as an argument to matches(). In matches(), for each Criterion object iterated over in the for loop, the code retrieves the corresponding Answer object in the answers HashMap, which is why you have to add an appropriate Answer to the Profile Project.

```
@Test
        public void matchAnswersFalseWhenMustMatchCriteriaNotMet() {
          Profile profile = new Profile("Bull Hockey, Inc.");
          Question question = new BooleanQuestion(1, "Got bonuses?");
          Answer profileAnswer = new Answer(question, Bool.FALSE);
          profile.add(profileAnswer);
          Criteria criteria = new Criteria();
          Answer criteriaAnswer = new Answer(question, Bool.TRUE);
          Criterion criterion = new Criterion(criteriaAnswer, Weight.MustMatch);
          criteria.add(criterion);

          boolean matches = profile.matches(criteria);
          assertFalse(matches);
        }
```

Notice that the test passed.



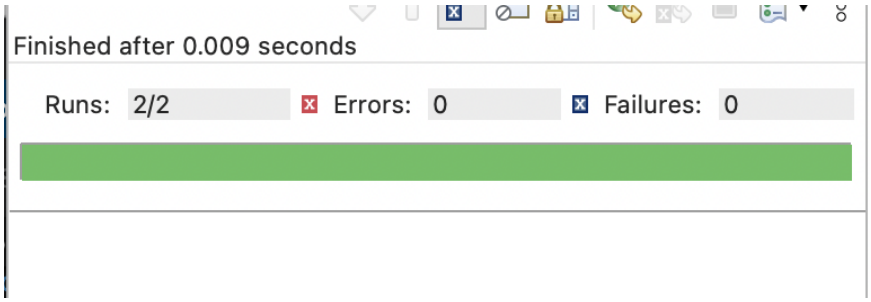
3- Creating a Second test

It seems that match gets set to true when the criterion weight is DontCare. The code below suggests that matches() should return true if a sole criterion sets match to true.

```
@Test
        public void matchAnswersTrueForAnyDontCareCriteria() {
          Profile profile = new Profile("Bull Hockey, Inc.");
          Question question = new BooleanQuestion(1, "Got milk?");
          Answer profileAnswer = new Answer(question, Bool.FALSE);
          profile.add(profileAnswer);
          Criteria criteria = new Criteria();
          Answer criteriaAnswer = new Answer(question, Bool.TRUE);
          Criterion criterion = new Criterion(criteriaAnswer, Weight.DontCare);
          criteria.add(criterion);

          boolean matches = profile.matches(criteria);
          assertTrue(matches);
        }
```

Notice that the test passed.



4- Initializing Tests with @Before Methods

Modify the ProfileTest.java to use @Before methods.

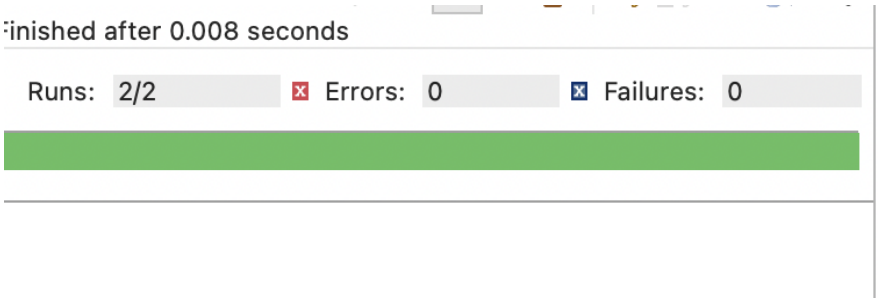
The first thing to look at is common initialization code in all (both) of the tests in ProfileTest. If both tests have such duplicate logic, move it into an @Before method.

```
public class ProfileTest {
        private Profile profile;
        private BooleanQuestion question;
        private Criteria criteria;

        @Before
        public void create() {
                profile = new Profile("Bull Hockey, Inc.");
                question = new BooleanQuestion(1, "Got bonuses?");
                criteria = new Criteria();
        }

        @Test
        public void matchAnswersFalseWhenMustMatchCriteriaNotMet() {
                Answer profileAnswer = new Answer(question, Bool.FALSE);
                profile.add(profileAnswer);
                Answer criteriaAnswer = new Answer(question, Bool.TRUE);
                Criterion criterion = new Criterion(criteriaAnswer, Weight.MustMatch);
                criteria.add(criterion);
                boolean matches = profile.matches(criteria);
                assertFalse(matches);
        }
        @Test
        public void matchAnswersTrueForAnyDontCareCriteria() {
                Answer profileAnswer = new Answer(question, Bool.FALSE);
                profile.add(profileAnswer);
                Answer criteriaAnswer = new Answer(question, Bool.TRUE);
                Criterion criterion = new Criterion(criteriaAnswer, Weight.DontCare);
                criteria.add(criterion);
                boolean matches = profile.matches(criteria);
                assertTrue(matches);
              }
        }
```

Notice that the test passed.



Question 1: If the JUnit chooses to run matchAnswersTrueForAnyDontCareCriteria() first, what is the sequence of events?

Question 2: In order to minimize the impact any one test has on another (avoiding static fields in test cases as well), create a more condensed but more readable arrange portion of each test by inlining some local variables.

Submissions:

Please submit

1- Screenshots of your code as well as any code you modify
2- Detailed ansers of the two questions above